

November 2008

Revised version

January 2009

`carlomat`
**A program for automatic computation of
lowest order cross sections**

Karol Kołodziej^{1,2}

*Institute of Physics, University of Silesia
ul. Uniwersytecka 4, PL-40007 Katowice, Poland*

Abstract

The current version of `carlomat`, a program for automatic computation of the lowest order cross sections of multiparticle reactions, is described. The program can be used as the Monte Carlo generator of unweighted events as well.

¹Supported by the Polish Ministry of Scientific Research and Information Technology as a research grant No. N N519 404034 in years 2008–2010 and by European Community’s Marie-Curie Research Training Network under contracts MRTN-CT-2006-035482 (FLAVIANet) and MRTN-CT-2006-035505 (HEPTOOLS).

²E-mail: karol.kolodziej@us.edu.pl

PROGRAM SUMMARY

Title of program: carlomat

Version: 1.0 (November 2008)

Catalogue identifier:

Program obtainable from: CPC Program Library or on request by e-mail from the author

Licensing provisions: none

Computers: all

Operating systems: Linux

Programming language used: FORTRAN 90/95

Distribution format: gzipped tar archive

Keywords: lowest order multiparticle reactions, Standard Model, automatic calculation of cross sections, Monte Carlo, event generation

Nature of physical problem

Description of two particle scattering reactions with possibly up to 10 particles in the final state with a complete set of the Feynman diagrams in the lowest order of the Standard Model.

Method of solution

The matrix element for a user specified process and phase space parametrizations, which are necessary for the multi-channel Monte Carlo integration of the lowest order cross sections and event generation, are generated automatically. Both the electroweak and quantum chromodynamics lowest order contributions are taken into account. Particle masses are not neglected in the program. Matrix elements are calculated numerically with the helicity amplitude method. Constant widths of unstable particles are implemented by modifying mass parameters in corresponding propagators.

Restrictions on complexity of the problem

The number of external particles is limited to 12. Only the Standard Model is implemented at the moment in the program. No higher order effects are taken into account, except for assuming the fine structure constant and the strong coupling at appropriate scale and partial summation of the one particle irreducible loop corrections by introducing fixed widths of unstable particles.

Typical running time

Generation of the Fortran code with `carlomat` on a PC with the Pentium 4 3.0 GHz processor for reactions with 8 particles in the final state relevant for the associated associated top quark pair and Higgs boson production and decay takes about 10 minutes CPU time. This relatively long time of the code generation is determined by a lot of write to and read from a

disk commands which have to be introduced in order to circumvent limitations of the Fortran compilers concerning possible array sizes. The compilation time of generated routines depends strongly on a compiler used and an optimization option chosen. Typically, for the reactions mentioned, it takes about one hour to compile all the routines generated. Most of the time is used for the compilation of the kinematical routines. The execution time of the Monte Carlo (MC) integration with about 2 million calls to the integrand amounts typically to a few hours and, if the MC summing over polarizations is employed, it is dominated by computation of the phase space normalization.

LONG WRITE-UP

1 Introduction

Many interesting aspects of the Standard Model (SM) and models beyond it can be studied through investigation of reactions involving a few heavy particles at a time whose observation becomes possible owing to the increasing collision energy and luminosity of particle colliders, such as, e.g., the Large Hadron Collider (LHC), or the International Linear Collider (ILC) [1]. As the heavy particles are usually unstable, they almost immediately decay leading to reactions with several light particles in the final state which receive contributions typically from many thousands of Feynman diagrams already in the lowest order of SM. Although, in their overwhelming majority, those diagrams constitute background to the “signal diagrams” of production and decay of the heavy particles one wants to investigate, they must be all taken into account in reliable SM predictions for such reactions. This can be done practically only through a fully automated calculational process.

To be more specific, let us consider, e.g. a reaction of associated production of a top quark pair and a Higgs boson at the ILC

$$e^+e^- \rightarrow t\bar{t}H \quad (1)$$

that can be used to measure the top-Higgs Yukawa coupling [2]. The top and antitop decay, even before they hadronize, predominantly into bW^+ and $\bar{b}W^-$, respectively, the electroweak (EW) bosons subsequently decay into a fermion-antifermion pair each and the Higgs boson, if it has mass $m_H < 140$ GeV, which is favoured by the direct searches at LEP and theoretical constraints in the framework of the SM, decays mostly into a $b\bar{b}$ -quark pair. Thus, reaction (1) will actually be detected through reactions of the form

$$e^+e^- \rightarrow b\bar{b}b\bar{b}f_1\bar{f}_1f_2\bar{f}_2, \quad (2)$$

where $f_1, \bar{f}_1 = \nu_e, \nu_\mu, \nu_\tau, u, c$ and $f_2, \bar{f}_2 = e^-, \mu^-, \tau^-, d, s$ are the decay products of the W -bosons coming from decays of the t - and \bar{t} -quark. Except for 20 ‘signal’ Feynman diagrams of the associated production of a top quark pair and a Higgs boson the representatives of which

are depicted in Fig. 1, there are many other, typically $\mathcal{O}(10^4)$, off resonance background diagrams which contribute to any of reactions (2). For example, taking into account both the

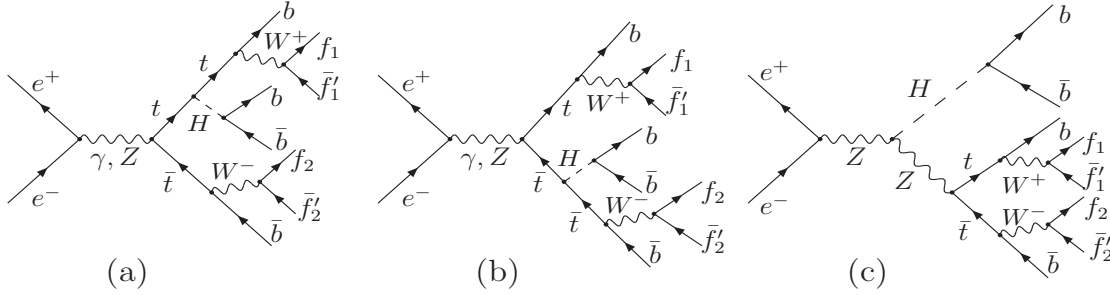


Figure 1: Representative signal Feynman diagrams of reaction (2) in the unitary gauge. The remaining diagrams are obtained by all possible permutations of the two b and two \bar{b} lines. The Higgs boson coupling to electrons has been neglected.

EW and quantum chromodynamics (QCD) lowest order contributions in the unitary gauge, neglecting of the Yukawa couplings of the fermions lighter than c quark and τ lepton, reactions

$$e^+e^- \rightarrow b\bar{b}b\bar{b}\tau^+\nu_\tau\mu^-\bar{\nu}_\mu, \quad (3)$$

$$e^+e^- \rightarrow b\bar{b}b\bar{b}u\bar{d}\mu^-\bar{\nu}_\mu, \quad (4)$$

$$e^+e^- \rightarrow b\bar{b}b\bar{b}u\bar{s}d\bar{c}, \quad (5)$$

$$e^+e^- \rightarrow b\bar{b}b\bar{b}u\bar{d}d\bar{u} \quad (6)$$

receive contributions from 21 214, 26 816 and 39 342 and 185 074 Feynman diagrams, respectively. To which extent the background contributions may affect the associated production of the top quark pair and Higgs boson in some of the reactions listed above has been discussed in [3].

Although there exists several multipurpose Monte Carlo (MC) generators such as HELAC/PHEGAS [4], AMEGIC++/Sherpa [5], O'Mega/Whizard [6], MadGraph/MadEvent [7], ALPGEN [8], CompHEP/CalcHEP [9], or recently released Comix [10], three years ago, when work on `carlomat` started, none of them was able to handle multiparticle reactions such as (3)–(5), not to mention reaction (6). This situation has changed quite recently and now some of the publicly available MC generators, as e.g. O'Mega/Whizard or HELAC/PHEGAS, are able to deliver complete SM predictions for reactions with 8 and more particles in the final state, including both the EW and QCD contributions in the leading order. However, when first physics results obtained with `carlomat`, on the off resonance background in (1) [3], were submitted for publication in spring 2008, they could only be partially checked against generators that were publicly available then.

The main aim of `carlomat`, a new multipurpose program for automatic computation of the lowest order cross sections, is to provide the reliable description of multiparticle reactions. This is a very challenging task, not only because of complicated matrix elements, involving many thousands of the Feynman diagrams, but also because of the necessity of performing integration over a multidimensional phase space. Not only may `carlomat` be useful for tests

of the complicated calculations done with other, already well established, generators. It may also offer the user other advantages, such as a relatively high speed of the MC computation, an easy control over mappings of peaks in the integrand, a possibility of calculating polarized cross sections, or a possibility of easy hand made modifications due to the use of traditional Feynman diagram approach. This write-up describes the current version of the program.

2 Basics features of the program

`carlomat` is a program written in Fortran 90/95. It generates the matrix element for a user specified process and phase space parametrizations which are later used for the multichannel MC integration of the lowest order cross sections and event generation. The program takes into account both the EW and QCD lowest order contributions. Particle masses are not neglected in the program. The number of external particles is limited to 12 and only the SM is currently implemented in the program.

`carlomat` works according to the following scheme. The user specifies the process he/she wants to have calculated. Then topologies for a given number of external particles are generated and checked against Feynman rules which have been coded in the program. In this process, helicity amplitudes, the colour matrix and phase space parametrizations are generated. Finally, they are copied to another directory where the numerical program can be executed.

2.1 Generation of topologies

Let us consider models with triple and quartic couplings. Generation of topologies starts with 4 topologies of a 4 particle process which are depicted in Fig. 2. The 25 topologies of a 5 particle process are obtained by attaching line No. 5 to each line, including the internal ones, and to each triple vertex of the graphs in Fig. 2. Particles No. 6, 7 and so on, are added

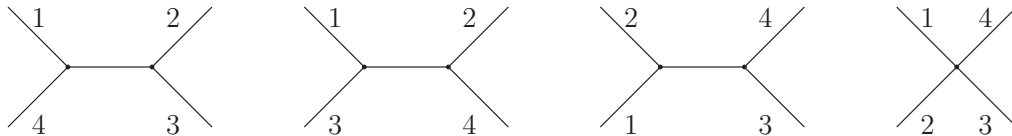


Figure 2: The topologies of a 4 particle process.

recursively in the same way. The number of topologies grows dramatically with the number of external particles as can be seen from Table 1. However, for a process with n external particles, it is enough to generate topologies for $n - 1$ particles and then, while adding the n -th particle, to check whether a topology results in a Feynman diagram or not. Topologies can be generated and stored on a disk prior to the program execution.

Table 1: No. of topologies for a given No. of particles.

No. of particles	No. of topologies
6	220
7	2 485
8	34 300
9	559 405
10	10 525 900
11	224 449 225

2.2 Feynman diagrams

Actual initial and final state particles are assigned to lines $1, 2, 3, \dots, n$ in a strict order. Each topology is divided into two parts which are separately checked against the Feynman rules. While doing so, two, or three, external lines are joined by means of a triple, or quartic, vertex of the implemented model, respectively. In this way a new off shell particle is formed. Another particle is formed by adding either the next external line, or an off shell particle that has been already formed. At the stage of matrix element generation, such particles are represented in `carlomat` by the derived data type called `particle` that is defined in the following way

```
type particle
integer :: wr,ct
integer :: k(4)
integer, pointer :: ip(:)
end type particle,
```

where

`wr`=1/0 if the particle has already/not yet been written to a file,

`ct`=1/3/8 for a colour singlet/triplet/octet state,

`k(1)` – the number of the vertex that has been used to form the particle,

`k(2),k(3),k(4)` – the numbers of particles that have been joined in the vertex, with `k(4)=0` for a triple vertex,

`ip(:)` – a list of all the off-shell particles that have been used in the definition of the current particle.

In the routine for computation of the matrix element, the particles being fermions, vector bosons, or scalar bosons are represented by spinors, polarization vectors, or just complex numbers, respectively.

The off shell particles and/or external particles are joined in this way until the two parts of a considered topology are completely covered. If they match into a propagator of the implemented model then the topology is accepted. Once the topology has been accepted, the

‘longer’ part of it is further divided so that the Feynman diagram is made of 3 or 4 parts, joint to form a triple or quartic vertex of the model. This reduces the number of different particles that must be defined for all the Feynman diagrams.

When the diagram is created, the corresponding particles are used to construct the helicity amplitude, colour factor (matrix) and phase space parametrization which are stored on the disk. Once all the topologies have been checked subroutines for calculating the matrix element, colour matrix and phase space integration are written.

2.3 Helicity amplitudes

Routines for computing the helicity amplitudes have been collected in a separate directory named `carlolib`. Most of the routines in `carlolib` had been already written for an MC program `eett6f` [11] for calculating lowest order cross sections of reactions $e^+e^- \rightarrow 6$ fermions, relevant for $t\bar{t}$ -pair production and decay, but they have been improved and tailored to meet the needs of the automatic generation of amplitudes in `carlomat`. In order to speed up the computation the MC summing over helicities has been implemented in the program. To illustrate how it works let us consider a sum over two polarization states $\lambda = \pm 1$

$$\sum_{\lambda=\pm 1} M^*(\lambda) M(\lambda) = \sum_{\lambda=\pm 1} u_a^\dagger(\lambda) \mathcal{O}_a^\dagger \mathcal{O}_b u_b(\lambda), \quad (7)$$

where the polarized amplitude that depends on λ has been written as

$$M(\lambda) = \mathcal{O}_a u_a(\lambda). \quad (8)$$

In Eq. (8), $u(\lambda) \equiv u(\mathbf{p}, \lambda)$ is the spinor representing a particle or an antiparticle of momentum \mathbf{p} and helicity $\lambda/2$, \mathcal{O} stands for the rest of amplitude $M(\lambda)$ that does not depend on λ and a summation over the Dirac index $a = 1, 2, 3, 4$ has been assumed. Now let us define the spinor

$$w(\alpha) = e^{i\alpha} u(+1) + e^{-i\alpha} u(-1), \quad (9)$$

with α being a random number uniformly distributed in the interval $[0, 1]$. Then the sum over polarizations can be replaced with the integral

$$\sum_{\lambda=\pm 1} M^\dagger(\lambda) M(\lambda) = \frac{1}{2\pi} \int_0^{2\pi} w_a^\dagger(\alpha) \mathcal{O}_a^\dagger \mathcal{O}_b w_b(\alpha) d\alpha. \quad (10)$$

Indeed,

$$\begin{aligned} & \frac{1}{2\pi} \int_0^{2\pi} w_a^\dagger(\alpha) \mathcal{O}_a^\dagger \mathcal{O}_b w_b(\alpha) d\alpha = \\ & \frac{1}{2\pi} \int_0^{2\pi} \left(e^{-i\alpha} u_a^\dagger(+1) + e^{i\alpha} u_a^\dagger(-1) \right) \mathcal{O}_a^\dagger \mathcal{O}_b \left(e^{i\alpha} u_b(+1) + e^{-i\alpha} u_b^\dagger(-1) \right) d\alpha = \\ & u_a^\dagger(+1) \mathcal{O}_a^\dagger \mathcal{O}_b u_b(+1) + u_a^\dagger(-1) \mathcal{O}_a^\dagger \mathcal{O}_b u_b(-1) = \sum_{\lambda=\pm 1} M^\dagger(\lambda) M(\lambda), \end{aligned}$$

$$\text{as } \int_0^{2\pi} e^{2i\alpha} d\alpha = \int_0^{2\pi} e^{-2i\alpha} d\alpha = 0 \text{ and } \int_0^{2\pi} d\alpha = 2\pi.$$

Analogously, a sum over three polarization states can be replaced with the integral according to

$$\sum_{\lambda=0,\pm 1} M^*(\lambda) M(\lambda) = \frac{1}{2\pi} \int_0^{2\pi} \epsilon_\mu^*(\alpha) \mathcal{O}^{\mu\dagger} \mathcal{O}^\nu \epsilon_\nu(\alpha) d\alpha, \quad (11)$$

where the longitudinal polarization component must be taken into account in the definition of $\epsilon(\alpha)$

$$\epsilon(\alpha) = \varepsilon(0) + e^{i\alpha} \varepsilon(+1) + e^{-i\alpha} \varepsilon(-1) \quad (12)$$

in terms of the polarization four vectors $\varepsilon(\lambda) \equiv \varepsilon(\mathbf{p}, \lambda)$, $\lambda = 0, \pm 1$, of a massive vector boson. Eq. (11) can be proven in exactly the same way as Eq. (10).

Although random sampling over external helicities increases the dimension of the integral, by one for each helicity, it does not actually spoil its convergence, as the dependence on α in (10) or (11) is rather smooth and the efficiency of the MC integration does not depend very much on the dimension of the integral. This has been explicitly checked for a number of reactions $e^+e^- \rightarrow 4 \text{ fermions } (\gamma)$ whose cross sections calculated with a program **ee4f γ** [12] that applies explicit summing over helicities have been successfully reproduced with **carlomat** using the random sampling over external helicities.

An explicit summing over helicities is also possible. While doing so, spinors or polarization vectors representing particles, both the on- and off-shell ones, are computed only once, for all the helicities of the external particles they are made of, and stored in arrays, which are later used in the sum over polarizations.

Possible poles in the propagators of unstable particles are regularized by their constant widths which are introduced through the complex mass parameters

$$\begin{aligned} M_B^2 &= m_B^2 - im_B \Gamma_B, & B = W, Z, H, \\ M_t &= \sqrt{m_t^2 - im_t \Gamma_t}, \end{aligned} \quad (13)$$

which replace masses in the corresponding propagators

$$\begin{aligned} \Delta_F^{\mu\nu}(q) &= \frac{-g^{\mu\nu} + q^\mu q^\nu / M_V^2}{q^2 - M_V^2}, \\ \Delta_F(q) &= \frac{1}{q^2 - M_H^2}, & S_F(q) = \frac{\not{q} + M_t}{q^2 - M_t^2}, \end{aligned}$$

both in the s - and t -channels. Propagators of a photon and gluon are taken in the Feynman gauge. The EW mixing parameter may be defined either to be real

$$\sin^2 \theta_W = 1 - \frac{m_W^2}{m_Z^2}, \quad (14)$$

with physical values of the W and Z boson masses, or complex

$$\sin^2 \theta_W = 1 - \frac{M_W^2}{M_Z^2}, \quad (15)$$

with M_W^2 and M_Z^2 given by Eq. (13). Using definitions (14), or (15), in the EW coupling constants is usually referred to as the fixed width scheme (FWS), or the complex mass scheme (CMS) [13]. The colour matrix is calculated only once at the beginning of execution of the numerical program after having reduced its size with the use of the SU(3) algebra properties. The calculation is performed numerically from the very definition, using the SU(3) structure constants and group generators in the fundamental representation [14].

2.4 Phase space integration

The integration over a multidimensional phase space is practically possible only with the use of Monte Carlo methods. A number of approaches has been developed in the literature to reduce the variance of the integrand [15]. The multichannel MC approach that is used in `carlomat` is described in this section.

A dedicated phase space parametrization is generated for each Feynman diagram of a process

$$p_1 + p_2 \rightarrow p_3 + \dots + p_n \quad (16)$$

with n external particles of four momenta p_i , $i = 1, 2, \dots, n$, in the centre of mass system. The Lorentz invariant phase space element of (16) is defined in a standard way

$$d^{3n_f-4} Lips = (2\pi)^4 \delta^{(4)} \left(p_1 + p_2 - \sum_{i=3}^n p_i \right) \prod_{i=3}^n \frac{d^3 p_i}{(2\pi)^3 2E_i}, \quad (17)$$

where $n_f = n - 2$ is the number of particles in the final state. The set of final state particles is divided, in a way that depends on a topology of the diagram, into subsets consisting of two elements each, an element being either an external particle or a subset of particles itself and use is made of the identity

$$\int ds_i \int \frac{d^3 q_i}{2E_i} \delta^{(4)}(q_i - q_{i_1} - q_{i_2}) = 1, \quad E_i^2 = s_i + \mathbf{q}_i^2.$$

Thus, Eq. (17) can be brought into the following form

$$d^{3n_f-4} Lips = (2\pi)^{4-3n_f} dl_0 dl_1 \dots dl_{n-4} ds_1 ds_2 \dots ds_{n-4}, \quad (18)$$

where dl_i , $i = 0, 1, \dots, n - 4$, is a two particle phase space element given by

$$dl_i = \frac{\lambda^{\frac{1}{2}}(s_i, q_{i_1}^2, q_{i_2}^2)}{2\sqrt{s_i}} d\Omega_i. \quad (19)$$

In Eq. (19), λ is the kinematical function, q_{i_1} and q_{i_2} are the four momenta of each particle (subsystem of particles) in subset i , which are defined in their relative centre of mass system, $\mathbf{q}_{i_1} + \mathbf{q}_{i_2} = \mathbf{0}$, Ω_i is the solid angle of momentum \mathbf{q}_{i_1} and invariants s_i are given by

$$s_i = \begin{cases} (q_{i_1} + q_{i_2})^2 = (E_{i_1} + E_{i_2})^2, & \text{for } i = 1, \dots, n-4 \\ (p_1 + p_2)^2 = s, & \text{for } i = 0 \end{cases}. \quad (20)$$

For processes with identical particles in the initial and final states, parametrization (18) must be slightly modified, by introducing the corresponding t invariants in a proper way. This has not been implemented in the program yet. Therefore, one may expect worse convergence of the MC integration when `carlomat` is used for such processes. The convergence is not much worse, however, if an angular cut of a few degree on the angle between the identical particles in the initial and final states is imposed, which has been tested for several reactions $e^+e^- \rightarrow 4$ fermions and $e^+e^- \rightarrow 6$ fermions with electrons and/or positrons in the final state.

Invariants s_i of Eq. (20) are randomly generated within their physical limits, s_i^{\min} and s_i^{\max} , which are automatically deduced from a topology of the Feynman diagram. They can be generated either according to the uniform distribution

$$s_i = (s_i^{\max} - s_i^{\min}) x_i + s_i^{\min}, \quad (21)$$

where x_i denotes a random variable uniformly distributed in the interval $[0, 1]$, or, if necessary, mappings of the Breit-Wigner shape of the propagators of unstable particles and $\sim 1/s$ behaviour of the propagators of massless particles are performed. An option is included in the program that allows to turn on the mapping if the particle decays into 2, 3, 4, ... on-shell particles. Different phase space parametrizations obtained in this way are important for testing purposes.

Phase space parametrizations (18) generated for each Feynman diagram

$$f_i(x) = d^{3n_f-4} Lips_i(x) \quad i = 1, \dots, N,$$

with N being the number of the diagrams and $x = (x_1, \dots, x_{3n_f-4})$ uniform random arguments, satisfy the normalization condition

$$\int_0^1 dx^{3n_f-4} f_i(x) = 1. \quad (22)$$

They are used to define a new multichannel probability distribution

$$f(x) = \sum_{i=1}^N a_i f_i(x), \quad (23)$$

with non negative weights a_i , $i = 1, \dots, N$, satisfying the condition

$$\sum_{i=1}^N a_i = 1 \quad (24)$$

which guaranties that the combined distribution $f(x)$ is normalized if every distribution $f_i(x)$ satisfies normalization condition (22). The actual MC integration is done with the random numbers generated according to probability distribution $f(x)$. An option has been included in the program that allows for reducing the actual number of kinematical channels used in the integration. It can be turned on by choosing `iopkch=1` in `carlocom.f`. Each kinematical channel is then called with the same set of random arguments x and of all the weights of the channels with the same phase space normalization $f_i(x)$ only one weight is kept nonzero.

The integration is performed iteratively. The first iteration starts with all the nonzero weights a_i equal to each other, or with the weights determined in the initial scan of the integrand according to the following formula

$$a_i = \sigma_i / \sum_{j=1}^N \sigma_j, \quad (25)$$

where σ_j denotes the cross section obtained with the i -th kinematical channel. An option has been included in the program that allows to choose whether the weights should be determined anew according to Eq. (25) after every iteration, or a fraction of old weights, i.e. determined in the scan or previous iterations, should be preserved and transfered to the next iteration. This is controlled by specifying the fraction `po` in `carlocom.f`, as described in Section 3.2.

3 Description of the program

The two basic parts of the program which allow for a generation of the Fortran code and the MC computation of the cross section of a user defined process are described in the following two subsections. The default values of the input parameters and options used in the program are those specified below.

3.1 Fortran code generation

The part of the program that is responsible for generating routines necessary for the execution of the MC program is stored in a directory `code_generation`. The main program is stored in a file `carlomat.f`. The user should specify the process he wants to have calculated by giving a value to a character variable `process` in `carlomat.f`, e.g.

```
process='e+ e- -> b b~ mu+ vm d u~'.
```

He may also choose the following options in `carlomat.f`:

Should the MC summing over polarizations be done, `imcs=1 (yes)/else (no)?`

`imcs=1`. *Recommended for multiparticle reactions.*

Take into account pure EW contributions only, `iewk=1(yes)/else(no)?`

`iewk=0`. *Recommended; takes into account both QCD and EW contributions.*

Discard the Higgs boson contributions, `ihgs=1(yes)/else(no)?`

`ihgs=0`. *Recommended; Higgs boson contributions are taken into account.*

Then the program calls two subroutines: `readpart` and `genmat`.

In `readpart`, the initial and final state particles are read from `process`. They are compared with the particles of the implemented model stored in a file `modpart.dat` and assigned the necessary characteristics which will be used during execution of the program. If all the particles match with particles in `modpart.dat` the possible interactions are read from a file `vertices.dat`. The numbers of vertices of different kind should not exceed the corresponding maximum numbers specified in a module file `maxvals.f`. Except for the maximum numbers of vertices:

`nffb=48`, the number of fermion–fermion–boson vertices,

`nbbb=6`, the number of triple boson vertices,

`nbbbb=8`, the number of quartic boson vertices

`maxvals.f` contains also values of other parameters which allow to control execution of the program:

`ncol=3`, the number of colors,

`mxprt=100`, the maximum number of particles in a model,

`mxep1=12`, the maximum number of external particles,

`mxip1=100000`, the maximum number of the off-shell particles which can be created,

`mxvpl=4`, the maximum number of similar vertices, such as $uu\gamma$, uuZ , uuh , or uug ,

`mxncnt=20`, the maximum number of continuation lines in a command,

`mxcmdl=1000`, the maximum number of lines in selected routines as, e.g., `mdkk*.f`, `mmodkk*.f`,

`mxcmdb=20`, the maximum number of blocks in selected routines as, e.g., `kinkk*.f`, relevant for the speed of their compilation,

`mxmap=4`, the maximum number of particles in a subsystem whose propagator should be mapped.

The module contains also the name of the current version of the program.

In `genmat`, topologies for the actual number of n_{ext} external particles are generated, if the corresponding file containing pregenerated topologies for $n = n_{\text{ext}} - 1$ particles could not have been found in a directory `code_generation`. This is done by consecutive calls to a subroutine `gentop` with the number of particles $n = 4, 5, \dots, n_{\text{ext}}$. As long as $n < n_{\text{ext}}$ the topologies are being generated and stored in a file named `topoln.dat`. When `gentop` is called for the last time, with $n = n_{\text{ext}}$, each time a new topology is created by attaching the last particle to a topology that was generated in the previous call and that has been now read from a file `topoln-1.dat`, another subroutine `checktop` is called, which checks it against Feynman rules in a way described in Section 2.2. While doing so, the off-shell particles are created which, if the topology has been accepted, are used to generate the helicity amplitudes, the colour

matrix and phase space parametrizations for all the Feynman diagrams corresponding to the topology.

After all the topologies have been inspected, the following files necessary for a computation of the matrix element and phase space integration are created and stored in the working directory.

- Modules:

`partcls.f`, containing information about the process, as, e.g., numbers of particles, diagrams, etc.,

`umodkk.f`, a collection of complex arrays representing spinors, polarization vectors and scalars which are the building blocks of the diagrams,

`hmodkk.f`, a collection of integer arrays which determine a sequence of elements in arrays of `umodkk`, relevant if no MC summing over polarizations is done,

`pmodkk.f`, a collection of particle four momenta,

`mmodkk*.f`, a collection of the amplitudes.

- Subroutines:

`matrixel.f`, the squared matrix element averaged (summed) over polarizations and colors of the initial (final) state particles,

`mpol2.f`, the polarized squared matrix element averaged (summed) over colors of the initial (final) state particles,

`mtel*.f`, a collection of calls to routines of `carlolib` which calculate building blocks of the Feynman diagrams,

`mdkk*.f`, a collection of calls to routines of `carlolib` which calculate amplitudes of the Feynman diagrams,

`hsubkk.f`, calculates integer arrays which determine a sequence of elements in arrays of `umodkk`, relevant if no MC summing over polarizations is done,

`colsqkk.f`, calculates colour factors,

`kinkk*.f`, a collection of `mxcmdb` different phase space parametrizations,

`kincls.f`, calls the kinematical routines `kinkk*` in order to generate the final state particle four momenta,

`psnkk.f`, calls phase space normalization routines `psub*`,

`psub*.f`, calls the kinematical routines `kinkk*` in order to calculate the phase space normalization.

- Parts of the code:

`partcls.dat`, some information about the external particle of the process, included in a subroutine `parfixkk`,

`mtdr*.f`, adds up the amplitudes which have the same colour factors, included in a subroutine `mpol2`.

Once the execution of `carlomat` ends they are copied to another directory named `mc_computation`, where the MC program can be executed.

3.2 MC computation

The part of the program that is responsible for the MC computation is stored in a directory `mc_computation`. The main program is stored in a file `carlocom.f`. In the latter, the user should supply the following data:

The centre of mass energies

`aecm=(/500.d0/),`

for which the cross section should be computed together with a parameter

`ne=1`

being a size of the array `aecm`. If, for example, the program should run for 3 centre of mass energies: 500 GeV, 800 GeV and 1 TeV then `aecm=(/500.d0,800.d0,1.d3/)` and `ne=3`. Moreover, the following options should be chosen in `carlocom.f`.

Should the Born cross section be calculated, `iborn=1(yes)/else(no)`, in `niter` iterations with `ncalli` calls per iteration, i.e. with `niter × ncalli` calls?

`iborn=1, recommended,`

`niter=10,`

`ncalli=10000.`

The switch `iborn` has been introduced, since the implementation of the initial state radiation within the structure function approach is envisaged. Its only meaningful value in the current version of the program is 1.

The fraction of old integration weights which should be transferred to the next iteration

`po=0.d0, recommended value from 0 to 0.5.`

If `po=0.d0` then integrations weights a_i are calculated anew according to Eq. (25) and it may happen that, because of a low weight determined in the previous iteration, the i -th channel will not be selected resulting in $a_i = 0$ for the next iteration. Thus the number of kinematical channels used in the next iterations will be effectively reduced.

Generate the unweighted events, `imc<0(no)/else(yes)?`

`imc=2.`

The probability distribution according to which the momenta of the final state particles are generated is limited either by the maximum of the total cross section, for `imc=0`, or, for `imc=1,2,...`, by the maxima of the cross section in each bin of distribution No. `imc`. See discussion of an option `idis` below.

Should the Born cross section be scanned with `nscan0` calls, `iscan0=1(yes)/else(no)`?

`iscan0=1,`
`nscan0=10.`

If `iscan0=1` then each kinematical channel is called with the same weight equal to $1/n_{\text{kin}}^{\text{tot}}$, where $n_{\text{kin}}^{\text{tot}}$ is the total number of the channels, and the cross section is calculated with `nscan0` calls. The value of the cross section is later used to calculate the weight, according to Eq. (25), with which that particular channel will contribute to the integral in the first iteration.

Should the number of kinematical channels be optimized, `iopkch=1(yes)/else(no)`?

`iopkch=0,` *for processes with large number of the Feynman diagrams `iopkch=1` is recommended.*

Discard kinematical channels which contribute less than `qw` of the dominant channel in the initial scan.

`qw=0.0d0.` *No channels are discarded; recommended.*

Choose the scheme, `ischeme=1(complex mass scheme)/else(fixed width scheme)`, see Eqs. (14) and (15) for the explanation,

`ischeme=1.`

Should distributions be calculated, `idis=1(yes)/0(no)`?

`idis=1.`

Variables, in which the distributions are to be calculated, the number of bins, etc., should be defined in a file `calcdis.f`.

Should cuts be imposed, `icuts=1(yes)/0(no)`?

`icuts=1.`

The cuts which can be controlled from the main program `carlocom` currently include

- angles: $\theta(l, \text{beam})$, $\theta(q, \text{beam})$, $\theta(g, \text{beam})$, $\theta(a, \text{beam})$, $\theta(a, l)$, $\theta(a, q)$, $\theta(q, q')$, $\theta(l, q)$, $\theta(l, g)$, $\theta(l, l')$
- energies: E_l , E_q , E_b , E_g , E_a
- invariant masses: $m_{ll'}$, $m_{qq'}$, $m_{bb'}$, $m_{jj'}$,

where l, l' are charged leptons, q, q' are quarks, b is a b -quark, g is a gluon, a is a photon and j is either a quark or a gluon.

The initial physical parameters: particle masses and widths, an inverse of the fine structure constant in the Thomson limit α , the Fermi coupling, the strong coupling constant α_s , number of colors and the conversion constant can be specified in module `inprms.f`. The module contains one more input parameter called `sqscut` that is an invariant mass cut, which is used in mappings of the photon and gluon propagators that couple to massless fermions. Its value

is relevant only if light fermion masses are chosen to be zero. If the width of the W boson, Higgs boson, or top quark is set to zero then the program will automatically replace it with a value calculated in the lowest order of SM.

The helicity amplitudes are calculated with the use of the routines which have been collected in a directory `carlolib`. The directory contains also other routines as `boost`, `kinff`, `kinin`, `lamsq`, `mapakk`, `mapbkk`, `mapskk`, which are used in the phase space calculation, `wwidth`, a routine for calculating the W -boson width in the lowest order of SM, or a random number generator `ranlux` [16].

4 Use of the program

At present `carlomat` is distributed as a single file `carlomat.tar.gz`. After executing command

```
tar -xzf carlomat.tar.gz
```

a working directory named `carlomat` with a few subdirectories is formed. The part of the program responsible for code generation is stored in subdirectory `code_generation`. After having specified a desired process in `carlomat.f` it can be run with a command

```
make test
```

executed in a command line in directory `code_generation`. Once ended, a file `test` is displayed on the screen that contains some details of the process under consideration, such as the external and off-shell particles, the number of Feynman diagrams, etc. and the generated files listed in Section 3.1 are moved to another working directory, named `mc_computation`, to which the user should change in order to perform the MC computation. After specifying the c.m.s. energy and the desired options in `carlocom.f` the MC program can be run by executing a command

```
make test.
```

The output of the run is stored in a file `test`. Both files named `test`, the one in a directory `code_generation` and the other in `mc_computation` should exactly reproduce files `test0` appended to both directories. If the MC events generation is turned on, i.e. for `imc=0,1,2,3,...`, then the generated events are stored in a file `mc_events.x`, with `x` being the c.m.s. energy. The data for each event are stored in `mc_events.x` in the following form:

```
# Event No. 1
p( 3)= 0.47046555E+02 -0.10891034E+02 0.42906506E+02 0.15190637E+02
p( 4)= 0.12373166E+03 0.59317748E+02 0.19678931E+02 0.10668003E+03
p( 5)= 0.25869511E+02 0.21415118E+02 -0.37079632E+01 -0.14030830E+02
p( 6)= 0.17716225E+03 -0.40997709E+02 0.81425324E+01 -0.17216082E+03
p( 7)= 0.65263801E+02 -0.52555950E+02 -0.20628053E+02 0.32737122E+02
p( 8)= 0.60926224E+02 0.23711829E+02 -0.46391953E+02 0.31583866E+02
```


where $\mathbf{p}(3), \dots, \mathbf{p}(8)$ are four momenta of the final state particles in the centre of mass system in exactly the same order as they appear in `process`. The first component is the energy and the other 3 components are the x, y and z components of the four momentum. The number of accepted events substantially increases when the program is run the second time with exactly the same choice of options and parameters. In each run, the maximum values of the cross section are stored in a file named `csmmax*` in directory `mc_computation` and they are automatically read from the disc in the next run of the program. The efficiency of events acceptance is not high, but it can be improved by a proper selection of the differential cross section and an appropriate choice of the number of bins.

5 Outlook

The following improvements of the program are envisaged in the near future.

- Dedicated treatment of soft and collinear external photons, as well as t -channel photon/gluon exchange.
- Interfaces to parton density functions, or the initial state radiation within the structure function approach.
- More efficient calculation of colour coefficients.

Moreover, interfaces to parton shower and hadronization programs should be worked on. Extensions of SM can be implemented and the corresponding lowest order cross sections can be calculated in a fully automatic way. Leading SM radiative corrections can be implemented, if the necessary subroutines are provided.

References

- [1] J. Brau, Y. Okada, N. Walker, et al., arXiv:0712.1950;
J.A. Aguilar-Saavedra, et al., arXiv:hep-ph/0106315;
T. Abe, et al., arXiv:hep-ex/0106056;
K. Abe, et al., arXiv:hep-ph/0109166.
- [2] A. Djouadi, J. Kalinowski, P.M. Zerwas, Mod. Phys. Lett. **A7** (1992) 1765; Z. Phys **C54** (1992) 255.
- [3] K. Kołodziej, S. Szczypiński, Nucl. Phys. B801 (2008) 153.
- [4] A. Kanaki, C.G. Papadopoulos, Comput. Phys. Commun. 132 (2000) 306;
C.G. Papadopoulos, Comput. Phys. Commun. 137 (2001) 247;
C.G. Papadopoulos, M. Worek, Eur. Phys. J. C50 (2007) 843;
A. Cafarella, C.G. Papadopoulos, M. Worek, arXiv:0710.2427.

- [5] F. Krauss, R. Kuhn, G. Soff, JHEP 0202 (2002) 044;
T. Gleisberg, S. Höche, F. Krauss, A. Schälicke, S. Schumann, J.C. Winter, JHEP 0402 (2004) 056.
- [6] M. Moretti, T. Ohl, J. Reuter, arXiv:hep-ph/0102195-rev;
W. Kilian, T. Ohl, J. Reuter, arXiv:0708.4233.
- [7] T. Stelzer, W.F. Long, Comput. Phys. Commun. 81 (1994) 357;
F. Maltoni, T. Stelzer, JHEP 02 (2003) 027.
- [8] M.L. Mangano, M. Moretti, F. Piccinini, R. Pittau, A. Polosa, JHEP 0307 (2003) 001.
- [9] E. Boos et al. Nucl. Instrum. Meth. A534 (2004) 250;
A. Pukhov, arXiv:hep-ph/0412191.
- [10] T. Gleisberg, S. Höche, arXiv:0808.3674.
- [11] K. Kołodziej, Comput. Phys. Commun. 151 (2003) 339.
- [12] K. Kołodziej, F. Jegerlehner, Comput. Phys. Commun. 159 (2004) 106.
- [13] A. Denner, S. Dittmaier, M. Roth, D. Wackeröth, Nucl. Phys. B560 (1999) 33 and Comput. Phys. Commun. 153 (2003) 462.
- [14] S. Pokorski, “Gauge Field Theories” 2nd Edition, Cambridge University Press.
- [15] R. Kleiss, W.J. Stirling, S.D. Ellis, Comput. Phys. Commun. 40 (1986) 359;
R. Kleiss, R. Pittau, Comput. Phys. Commun. 83 (1994) 141;
T. Ohl, Comput. Phys. Commun. 120 (1999) 13;
P. Draggiotis, A. van Hameren, R. Kleiss, Phys. Lett. B 483 (2000) 124;
A. van Hameren, C.G. Papadopoulos, Eur. Phys. J. C25 (2002) 563.
- [16] M. Luscher, Comput. Phys. Commun. 79 (1994) 100;
F. James, Comput. Phys. Commun. 79 (1994) 111.